

Symmetric Cryptography Protocols

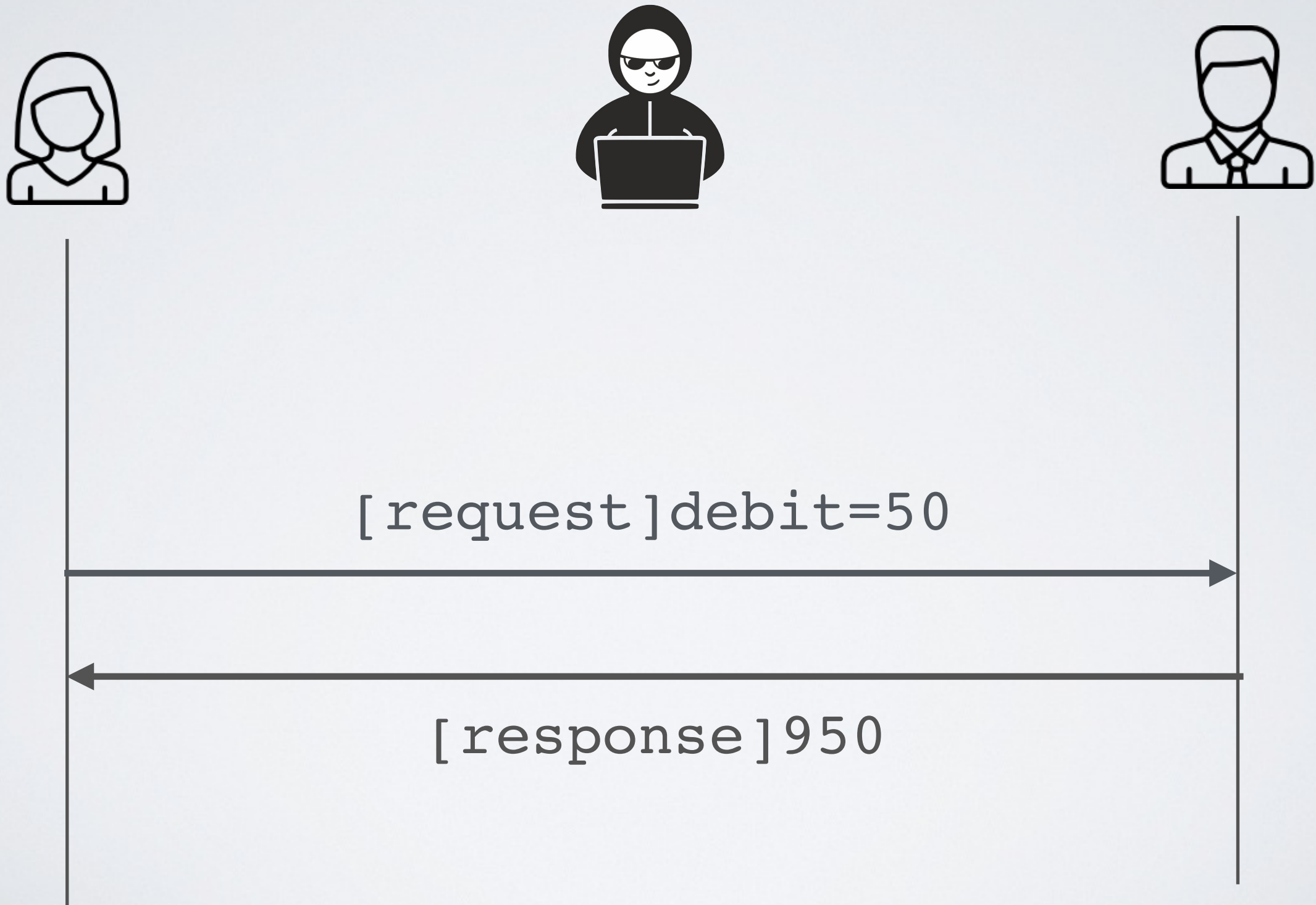
Thierry Sans

Security goals vs attacker's model



Let us consider **confidentiality, integrity and availability**

Example



Ensuring confidentiality with encryption



$E_k("[request]debit=50")$

tkS3bffBpdJvr96+mpLIAp0=

$D_k("tkS3bffBp...")$

$E_k("[response]950")$

tkS3b/LLuNVX1oLpww==

$D_k("tkS3b/LLu...")$

Ensuring integrity with an HMAC



$H_k(["request"]debit=50")$

`[request]debit=50`
`f89a73aa27f3ea6...`







$H_k(["request"]debit=50")$

$H_k(["response"]950")$

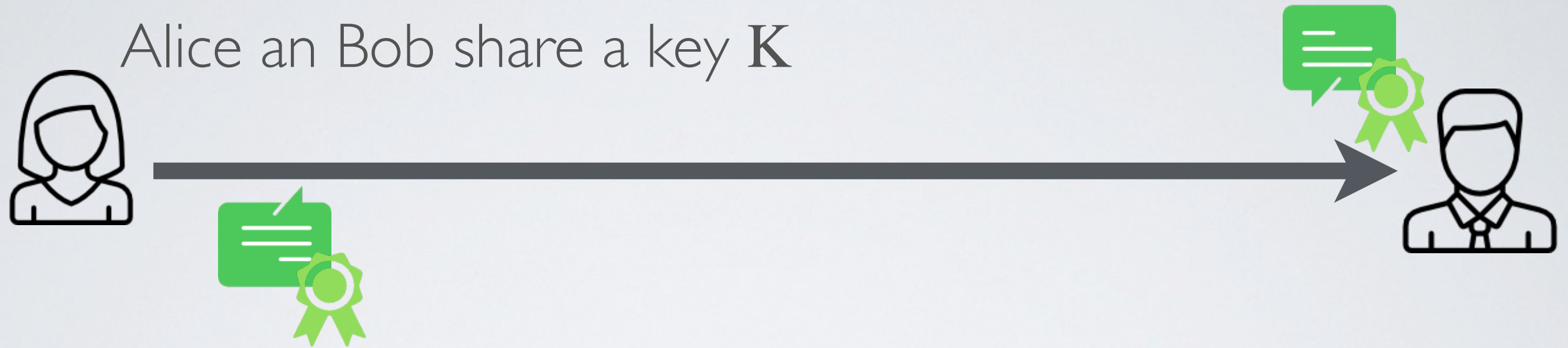
`[response]950`
`ee5a49c19fc252f...`

$H_k(["response"]950")$

Security mechanisms

	Encryption	MAC	Authenticated Encryption
Confidentiality			
Integrity			

Authenticated Encryption (2013)



Encrypt-and-MAC (E&M)

$$AE_K(m) = E_K(m) \parallel H_K(m)$$

e.g. *SSH*

MAC-then-Encrypt (MtE)

$$AE_K(m) = E_K(m \parallel H_K(m))$$

e.g. *SSL*

Encrypt-then-MAC (EtM)

$$AE_K(m) = E_K(m) \parallel H_K(E_K(m))$$

e.g. *IPsec*

Ensuring confidentiality and integrity with Authenticated Encryption



$AE_k(["request"]debit=50)$

30354WxPYF...

$AD_k("30354WxPYF...")$

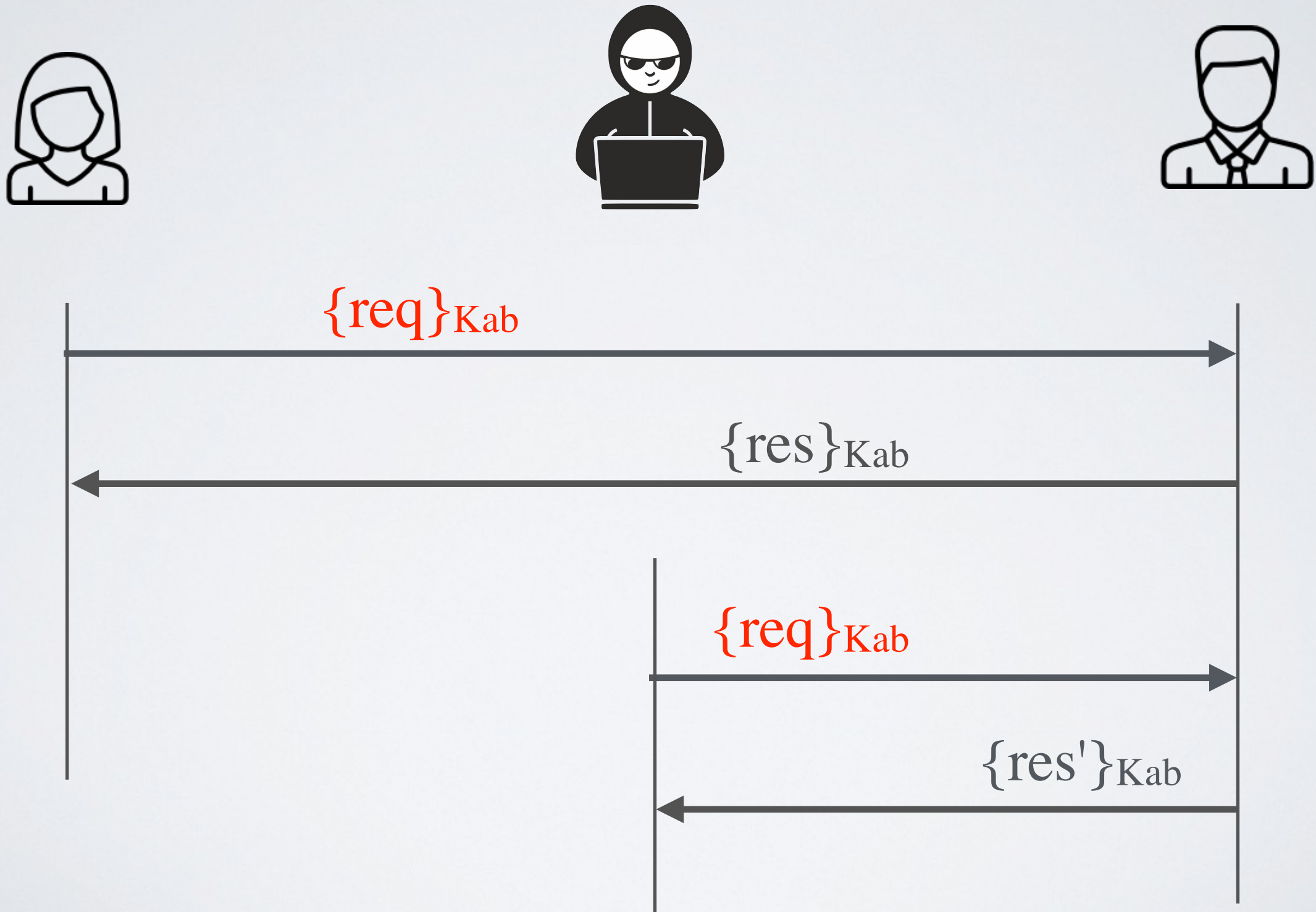
$AE_k(["response"]950)$

15qcK3Xcdwd ...

$AD_k("15qcK3Xcdwd...")$

Replay attacks

Replay attack

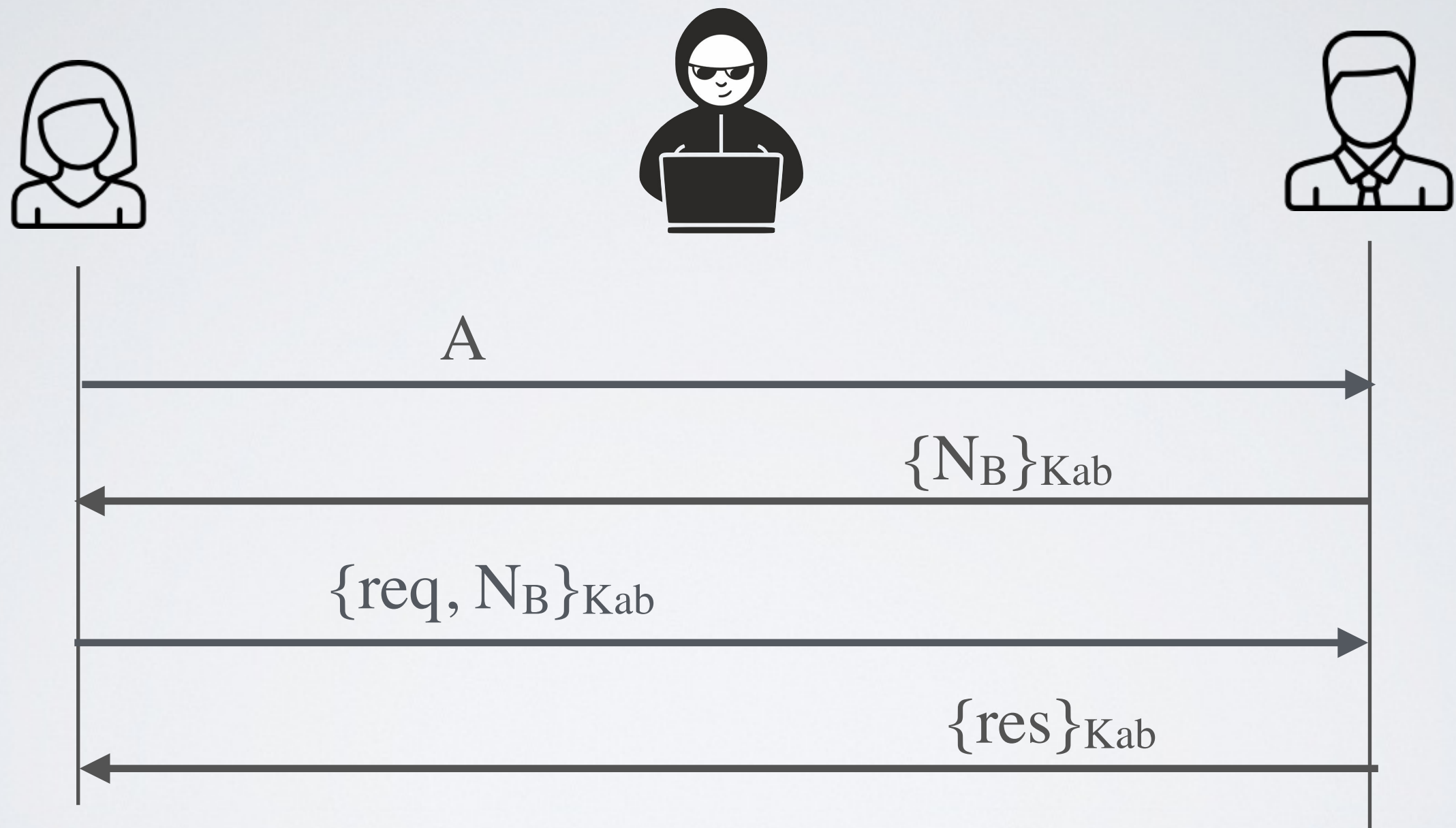


Counter replay attacks

Several solutions:

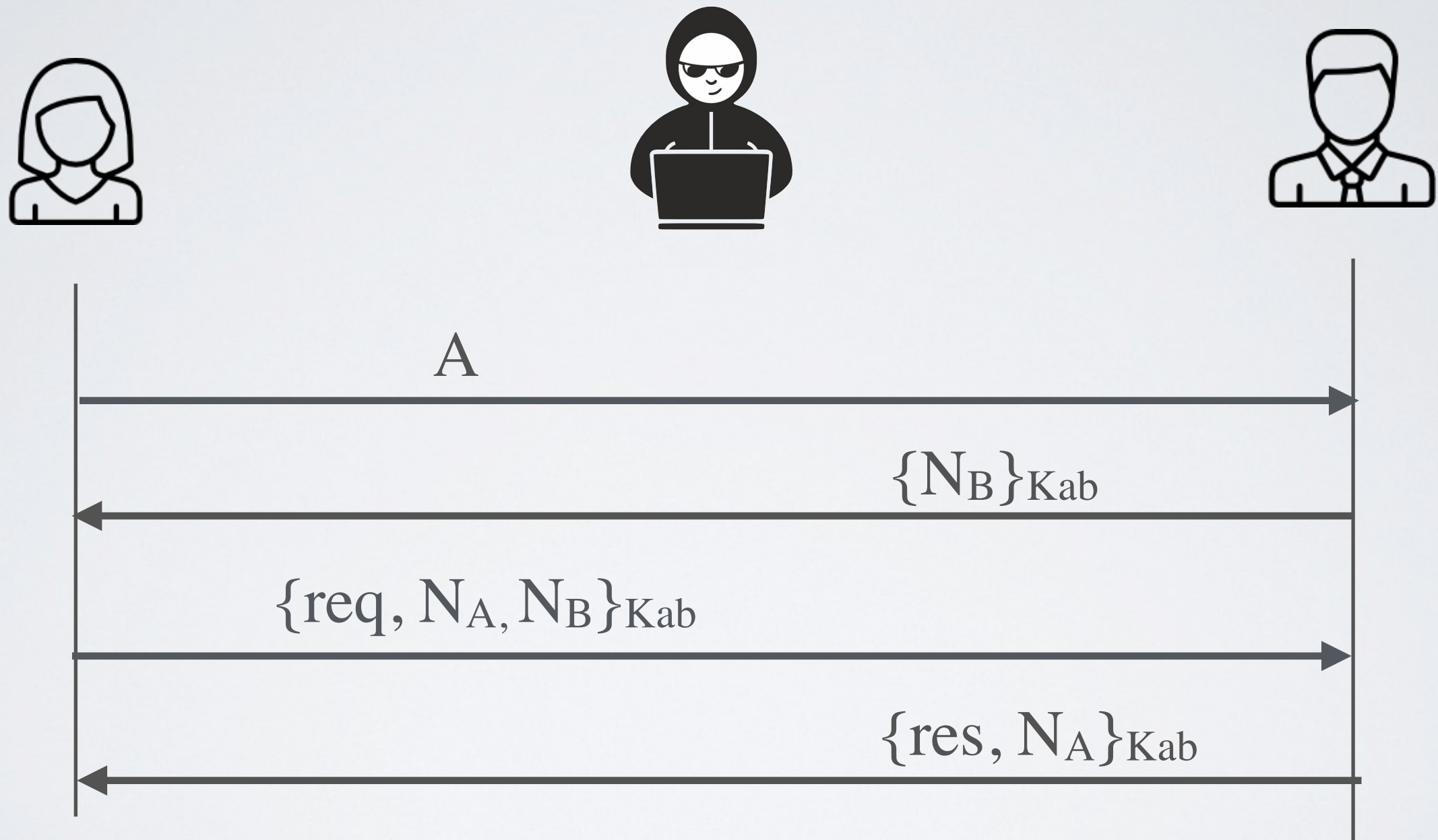
- use a nonce (random number)
- use sequence numbers
- use timestamps
- have fresh key for every transaction
(key distribution problem)

Defeat replay attack with a nonce (not fully secured)



Replay attack on the response!

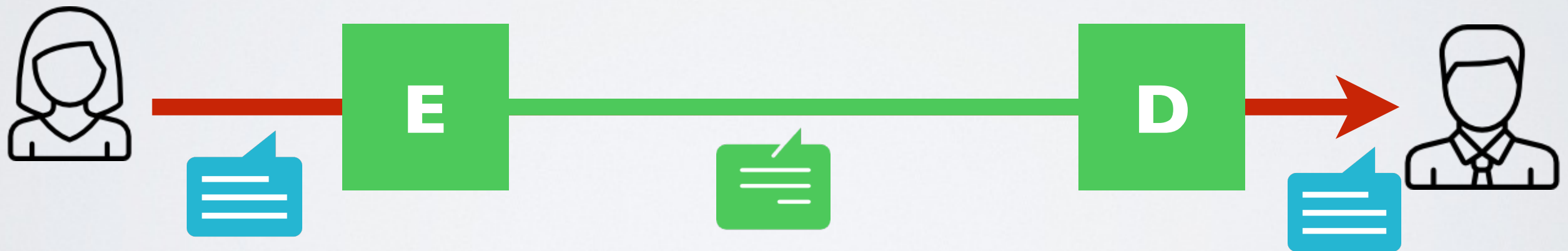
Defeat replay attack with a double nonce



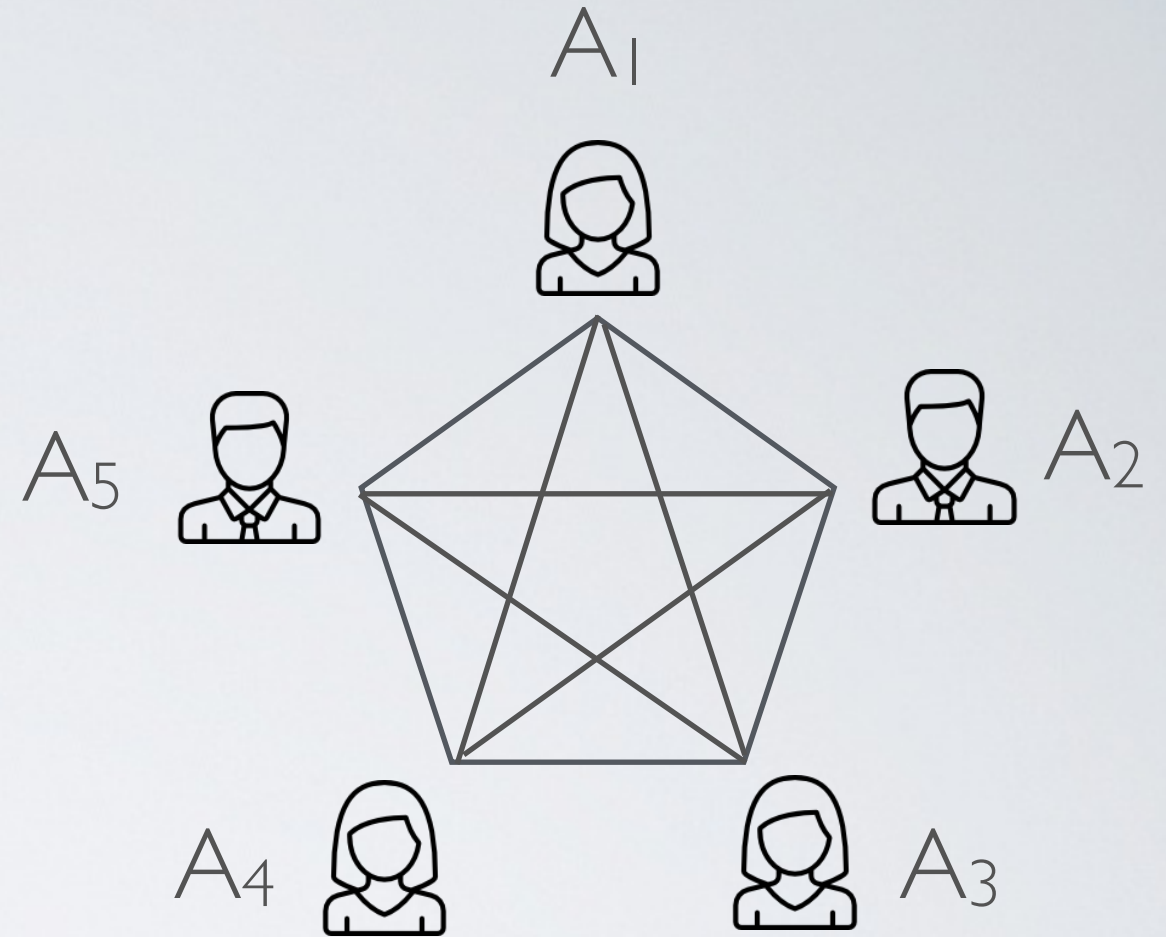
The challenge of key exchange

The big challenge with symmetric cryptosystems?

How do we **agree**
on the  ?



Naive Key Management

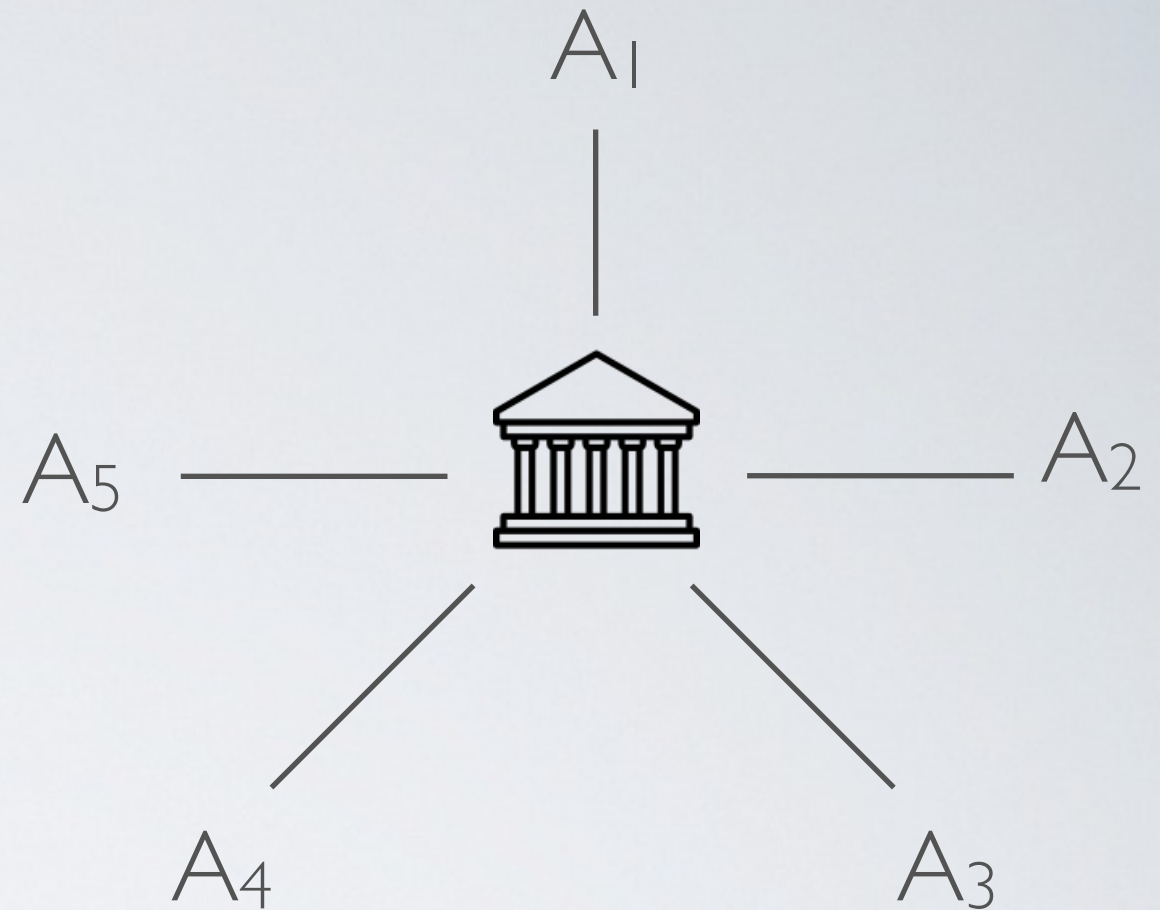


$A_1, A_2 \dots A_5$ want to talk

➡ Each pair needs a key : $n(n-1) / 2$ keys

⦿ Keys must be exchanged physically using a secure channel

(Better) centralized solution



$A_1, A_2 \dots A_5$ can talk to the KDC (Key Distribution Center)

- ➡ When A_i and A_j want to talk, the KDC can generate a new key and distribute it to them
- ⦿ We still have n keys to distribute somehow using a secure channel
- ⦿ The KDC must be trusted
- ⦿ The KDC is a single point of failure
- ➡ This is how *Kerberos* works

The Needham-Shroeder Symmetric Key Protocol

Assumptions

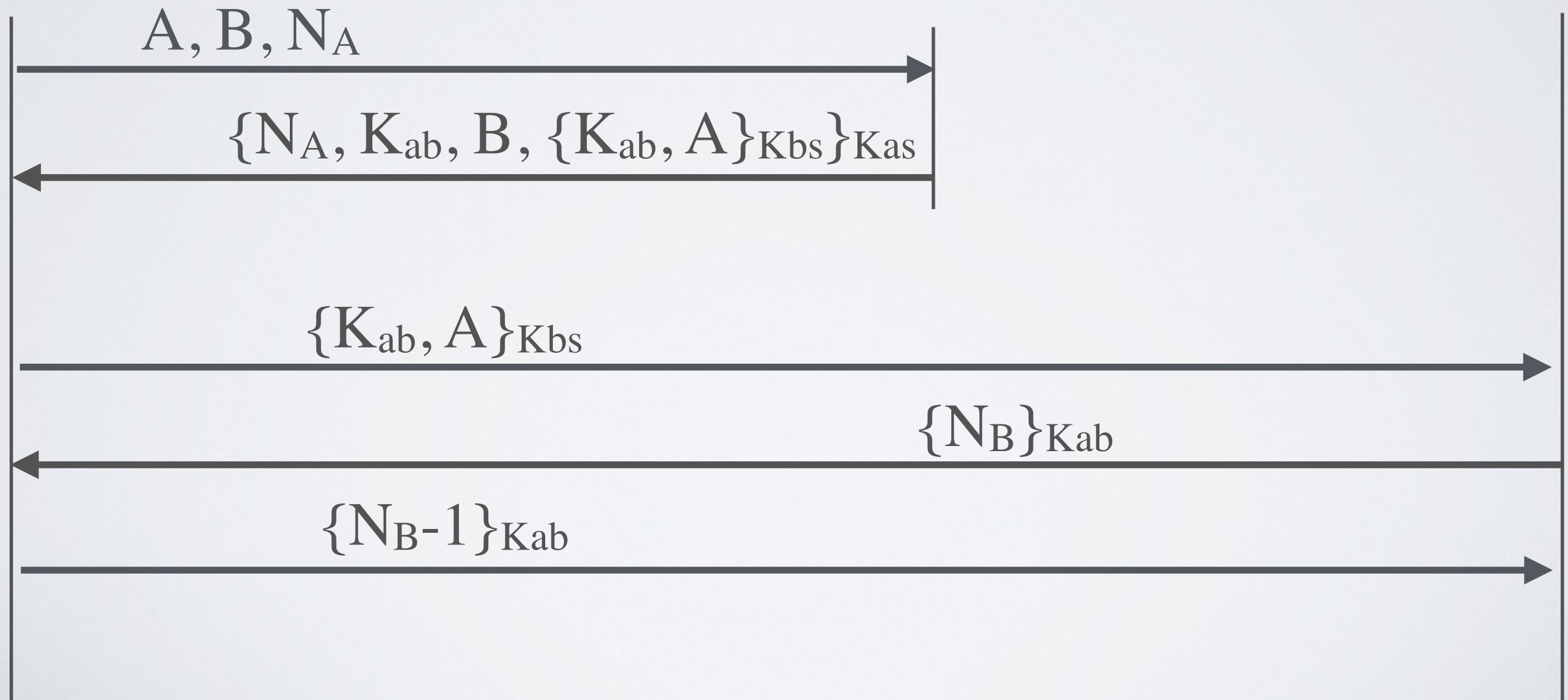
- 4 principals : Alice, Bob, Mallory, Key Distribution Server
- S shares a key with A, B and M respectively K_{as} , K_{bs} , K_{ms}
- A, B, M and S talk to each other using the same protocol

Goals

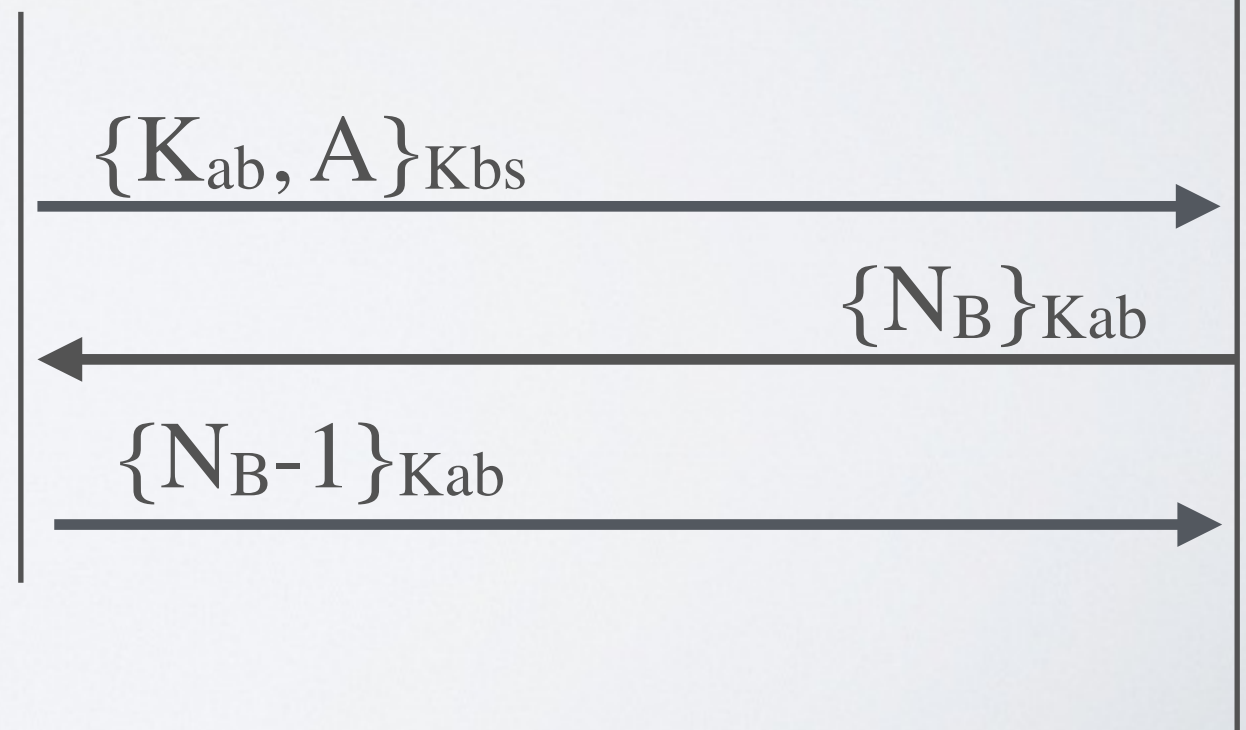
When two parties want to engage in the communication, they want to

1. make sure that they talk to the right person (authentication)
2. establish a session key

The vulnerable Needham-Shroeder Symmetric Key Protocol (1978)



Breaking the Needham-Shroeder Symmetric Key Protocol (1981)



Fixing the Needham-Shroeder Symmetric Key Protocol (1987)

